

DISTRIBUTED & MOBILE SYSTEMS

Eine Einführung

Autor: Walter Rothlin (22.10.21)

```
10 INPUT "Geben Sie bitte Ihren Namen ein"; A$
20 PRINT "Guten Tag, "; A$
30 INPUT "Wie viele Sterne möchten
35 S$ = ""
40 FOR I = 1 TO S
50 S$ = S$ + "*"
55 NEXT I
60 PRINT S$
70 INPUT "Möchten Sie noch mehr St
80 IF LEN(Q$) = 0 THEN GOTO 70
90 L$ = LEFT$(Q$, 1)
100 IF (L$ = "J") OR (L$ = "j") THEN GOTO 30
110 PRINT "Auf Wiedersehen";
120 FOR I = 1 TO 200
130 PRINT A$; " ";
140 NEXT I
150 PRINT
```

Sequenzen, if-goto

- Each application starts from scratch
- Maintenance intensive
- Migrationstests
- Reuse via cut / past
- Predefined functions as part of the language

```

# -----
# Name: umrechnungen.py
#
# Description: Rechnet verschiedene physikalische Grössen um.
#
# Autor: Walter Rothlin
#
# History:
# 26-Sep-2017   Walter Rothlin   Initial Version
# -----

doLoop = True
while doLoop:
    print(" Umrechnungen")
    print(" =====")
    print(" 1: Grad in Bogenmass")    # rad = grad*pi/180
    print(" 2: Bogenmass in Grad") # grad = rad*180/pi
    print()
    print(" 3: Fahrenheit in Celsius") #32F -> 0°C    100F -> 38.8°C    °C = (°F - 32) / 1.8
    print(" 4: Celsius in Fahrenheit") #32F -> 0°C    100F -> 38.8°C    °F = (°C * 1.8) - 32
    print()
    print(" 0: Schluss")

    antwort = input("\n Wähle:")
    if (antwort == "1"):
        gradValue=float(input("Grad:"))
        print(gradValue, "Grad ==> ", gradValue*3.141592/180, "Rad")

    if (antwort == "2"):
        radValue=float(input("Rad:"))
        print(gradValue, "Rad ==> ", radValue*180/3.141592, "Grad")

    if (antwort == "3"):
        pass

    if (antwort == "4"):
        pass

    if (antwort == "0"):
        doLoop = False

print("Ende....Done")

```

Loops and If-Then-Else
+ Better Maintenance

```

# -----
def grad2Rad(grad):
    return 3.1415*grad/180

def rad2Grad(rad):
    return 180*rad/3.1415

def fahrenheit2Celsius(fahrenheit):
    return (fahrenheit-32)/1.8

def celsius2Fahrenheit(celsius):
    return (celsius*1.8)+32

doLoop = True
while doLoop:
    VT52_cls_home()
    print("  Umrechnungen")
    print("  =====")
    print("  1: Grad in Bogenmass")    # rad = grad*pi/180
    print("  2: Bogenmass in Grad")  # grad = rad*180/pi
    print()
    print("  3: Fahrenheit in Celsius")  #32F -> 0°C    100F -> 38.8°C    °C = (°F - 32) / 1.8
    print("  4: Celsius in Fahrenheit")  #32F -> 0°C    100F -> 38.8°C    °F = (°C * 1.8) - 32
    print()
    print("  0: Schluss")

antwort = input("\n  Wähle:")
if antwort == "1":
    VT52_cls_home()
    print("Grad --> Bogenmass")
    gradValue=float(input("Grad:"))
    print("Grad={grad:1.2f} ==> Rad={rad:1.2f}".format(grad=gradValue,rad=grad2Rad(gradValue)))
    halt()

```

Own functions with parameters

+ Easier maintenance

+ Elimination of redundant code

```
from waltisLibrary import *
```

```
doLoop = True
while doLoop:
    VT52_cls_home()
    print("  Umrechnungen")
    print("  =====")
    print("  1: Grad in Bogenmass") # rad = grad*pi/180
    print("  2: Bogenmass in Grad") # grad = rad*180/pi
    print()
    print("  3: Fahrenheit in Celsius") #32F -> 0°C
    print("  4: Celsius in Fahrenheit") #32F -> 0°C
    print()
    print("  0: Schluss")

    antwort = input("\n Wähle:")
    if (antwort == "1"):
        VT52_cls_home()
        print("Grad --> Bogenmass")
        gradValue=float(input("Grad:"))
        print("Grad={grad:1.2f} ==> Rad={rad:1.2f}".format(grad=gradValue,rad=grad2Rad(gradValue)))
        halt()

    if (antwort == "2"):
        VT52_cls_home()
        print("Bogenmass --> Grad")
        radValue=float(input("Rad:"))
        print("Rad={rad:1.2f} ==> Grad={grad:1.2f}".format(rad=radValue,grad=rad2Grad(radValue)))
        halt()

    if (antwort == "3"):
        VT52_cls_home() # http://www.metric-conversions.org/de/temperatur/fahrenheit-in-celsius.htm
        print("Fahrenheit in Celsius")
        fahrenheitValue=float(input("Fahrenheit:"))
```

Function Libraries

- + Easier Maintenance
- + Reuse possible
- + Elimination of redundant code
- + Separation application / reusable libs


```
import math
import os
import sys
import time
import datetime

from time import sleep
```

```
# Primzahlen Functions
```

```
# -----
```

```
def isPrimzahl(aZahl):
    isPrim = True
    if (aZahl == 1):
        isPrim = True
    else:
        if (aZahl == 2):
            isPrim = True
        else:
            obergrenze = int((aZahl / 2) + 2)
            for i in range(2, obergrenze):
                if ((aZahl % i) == 0):
                    isPrim = False
    return isPrim

def getNextPrimzahl(zahl):
    aZahl = zahl + 1
    while (isPrimzahl(aZahl) == False):
        aZahl = aZahl + 1
    return aZahl
```

External Function Libraries

- + Reuse possible
- + Because of Internet
 - + open source
 - + sharing for free
- + IDE simplifies to import
- No states
- Language dependency

```

if (inTestMode):
    print("IN TEST-MODE")
    print("    time_T2                : ",time_T2)
    print("    schwellwert_H_OffImmediate: ",schwellwert_H_OffImmediate)
    print()
    print("=====\\n")
print(hsrKaelteMaschine.toString())
logFile.addLogEntry(hsrKaelteMaschine.toStringForLog())
if (hsrKaelteMaschine.isEmergencyOff_Active()):
    print("-----1")
    print("Emergency Off pressed!! Will not start!!!")
    hsrKaelteMaschine.doEmergencyStop()
    verdichter_1.doEmergencyStop()
    verdichter_2.doEmergencyStop()
    # t2State.setState_NotStarted() # Waterpump steys on in EmercencyOff
else:
    print("-----2")
    vorhandeneEnergie      = hsrKaelteMaschine.getExistingEnergy()
    hochdruck              = hsrKaelteMaschine.getHighPressure()
    niederdruck            = hsrKaelteMaschine.getLowPressure()
    waterTemp              = hsrKaelteMaschine.getWaterTemp_PT1000()
    envTemp                = hsrKaelteMaschine.getEnvironmentTemp_PT1000()

    # -----
    # Calculated values
    # -----
    lowPressureError        = niederdruckOff.setState(niederdruck,verbal = True)
    highPressureError        = hochdruckOff.setState(hochdruck ,verbal = True)
    pressureError = (lowPressureError or highPressureError)
    if (pressureError):
        if (t1State.isState_NotStarted()):
            t1 = Timer(time_T1, T1Over, args=[t1State])
            t1State.setState_Ticking()
            t1.start()
        print("pressureError      :",pressureError,end="")
        if (lowPressureError):
            print("    (lowPressureError)",end="")
            hsrKaelteMaschine.setShutOffValveToOpen(True)
            time.sleep(lowPressureErrorShutOffValveReopenTime)

```

Classes: Methods with states

- + Reuse improved (inheritance)
- + Decoppeling
- + Simple interfaces
- Language dependend
- No runtime distribution

```

# -----
import math
import time
import datetime

from waltisLibrary import *

class Verdichter:

    # Ctr (Konstruktor)
    # -----
    def __init__(self, name, pMin, pMax, outputTiefpass, hsrKaelteMaschine):
        self.name = name
        self.pMin = pMin
        self.pMax = pMax
        self.outputTiefpass = outputTiefpass
        self.hsrKaelteMaschine = hsrKaelteMaschine
        self.istSpeed = 0
        self.sollSpeed = 0
        self.verdichterOn = False
        self.isVerdichterOnState = False
        self.wiederEinschaltenMoeglich = False
        self.lastTimeOff = datetime.datetime.now()

    # Methoden (setter / getter)
    # -----
    def setValues(self, sollSpeed, emergencyOff, einschaltverzoeget):
        currTime = datetime.datetime.now()
        wiedereinschaltverzoegerung = "0:00:10"
        print("==> setValues ({x1:1s})::sollSpeed:{x2:4.0f}%".format(x1=self.name, x2=sollSpeed))

    def (emergencyOff):
        self.sollSpeed = 0
        self.verdichterOn = False
        self.resetRestartTime()
    else:
        if (sollSpeed > self.pMax):
            self.sollSpeed = self.pMax
            self.verdichterOn = True
        elif (sollSpeed < self.pMin):
            self.sollSpeed = 0
            if (self.verdichterOn):
                self.verdichterOn = False

```

Class-Libraries (Packages)

- + Reuse improved
- + Decoppeling
- + Simple interfaces
- Language dependend
- No runtime distribution


```
while True:
    responseStr = requests.get("https://api.openweathermap.org/data/2.5/v
    jsonResponse = json.loads(responseStr.text)

    temp = jsonResponse['main']['temp']
    pressure = jsonResponse['main']['pressure']
    humidity = jsonResponse['main']['humidity']
    lon = jsonResponse['coord']['lon']
    lat = jsonResponse['coord']['lat']
    cloud = jsonResponse['weather'][0]['description']

    print(temp, pressure, humidity, lon, lat, cloud)
    time.sleep(pollingTime)
```

- + simple reuse
- + nothing to install
- API des Services
- Functional programming

```
me = tello.Tello()
me.connect()
print(me.get_battery())
me.streamon()
```

+ Object Oriented

+ Hidden API from Service

```
while True:
    frame = me.get_frame_read().frame
    frame = cv.resize(frame, (480, 360), interpolation= cv.INTER_AREA)
    bat = me.get_battery()
    height = me.get_height()
    attitude = me.get_attitude()
    cv.putText(frame, ("Bat: " + str(bat)), (10, 50), cv.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0))
    cv.putText(frame, ("Alt: " + str(height)), (10, 70), cv.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0))
    cv.putText(frame, ("Att: " + str(attitude)), (10, 90), cv.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0))

    cv.imshow("Output", frame)
    k = cv.waitKey(1) & 0xFF
    if k == 27:
        me.land()
    if k == ord('w'):
        me.send_rc_control(0, mySpeed, 0, 0)
    if k == ord('s'):
        me.send_rc_control(0, -mySpeed, 0, 0)
```

Adressierung: Host- und Domain-Names

Jeder Computer in einem Netzwerk muss eindeutig bezeichnet sein!

→ IP Address:

↳ Eindeutige Nummer (4-stellig, jede Stelle 0..255: $255^4=4\text{Mia}$)

↳ z.B. 107.18.128.198

Nummern kann man sich schlecht merken! Namen bringen mehr Flexibilität!

→ Hostname / Domain-Name:

↳ Eindeutiger Name für einen Computer

↳ z.B. celera.credit-suisse.ch

2nd Level Domain

1st Level Domain

Hostname + Domain-Name = Full qualified hostname

Übersetzt IP-Adressen in Hostnames und umgekehrt: → DNS (Domain Name Service)

ipconfig /all

zeigt die eigene IP Adresse

nslookup

übersetzt IP ← → Hostname

Adressierung: Port

Auf einem Computer sind mehrere Zuhörer / Sprecher gleichzeitig aktiv. Der Port bestimmt, mit welchem ich sprechen will.

Dienste	Port	Protokoll
echo	7	TCP
sysstat	11	TCP
chargen	19	TCP
ftp-data	21	TCP
ssh	22	TCP
telnet	23	TCP
smtp	25	TCP
nameserver	42	TCP
whois	43	TCP
tacacs	49	UDP
dns-lookup	53	UDP
dns-zone	53	TCP
orac1-sqlnet	66	TCP
tftp	69	UDP
finger	79	TCP
http	80	TCP
http-1	81	TCP
kerberos	88	TCP
pop2	109	TCP
pop3	110	TCP
sunrpc	111	TCP
sqlserv	118	TCP

https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

0 to 1023

are the *well-known ports* or *system ports*

1024 to 49151

are the registered ports

e.g. 8080 alternate http

49152–65535 ($2^{15}+2^{14}$ to $2^{16}-1$)

contains dynamic or private ports

TCP/IP Packet

TCP
IP Header
Size: 20..60 Bytes

Version	IHL	Type of Service		Total Length				
Identification				Flags	Fragment Offset			
Time to Live		Protocol=6 (TCP)		Header Checksum				
Source Address								
Destination Address								
Options						Padding		
Source Port				Destination Port				
Sequence Number								
Acknowledgement Number								
Data Offset		U R G	A C K	P S H	R S T	S Y N	F I N	Window
Checksum				Urgent Pointer				
TCP Options						Padding		
TCP Data (max 64kB)								

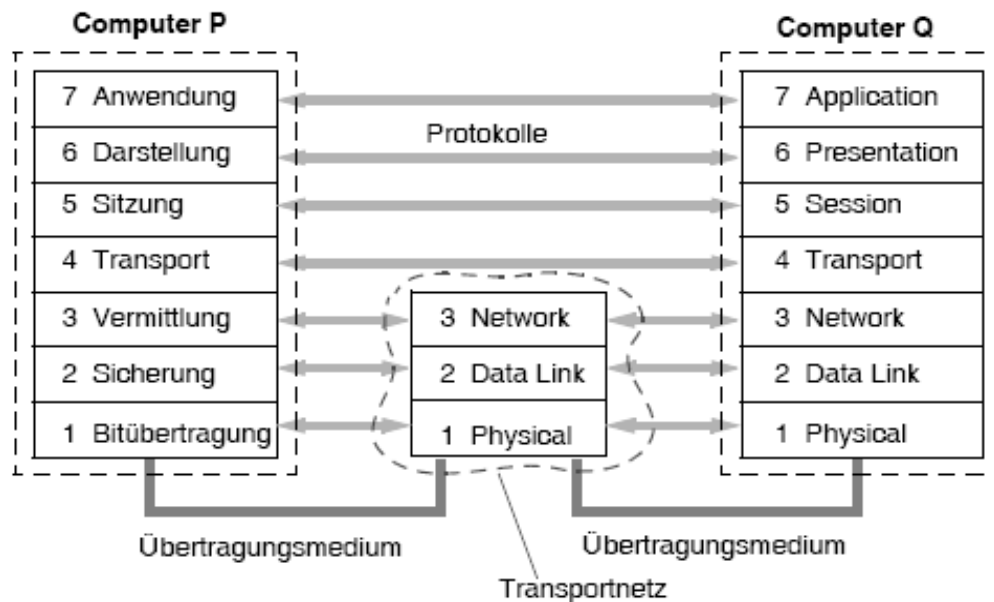
32-Bit / Zeile

IP-Adresse

z.B. 158.2.34.150

Port Nr: 0.. 65535

TCP/IP Open Systems Interconnect (OSI) Reference Model



7	Anwendungsschicht besteht aus den Anwendungen mit denen man das Netz nutzen kann
6	Darstellungsschicht standardisiert das Format der Daten auf dem Netz
5	Kommunikationssteuerungsschicht verwaltet die Verbindungen zwischen den Anwendungen
4	Transportschicht garantiert die fehlerfreie Datenübertragung durch Fehlererkennung und -korrektur
3	Vermittlungsschicht verwaltet die Verbindungen zwischen den Rechnern im Netz für die höheren Schichten
2	Sicherungsschicht sorgt für die zuverlässige Übertragung der Daten über die physikalischen Verbindungen
1	Bitübertragungsschicht definiert die physikalischen Eigenschaften der Übertragungswege

Client sends TCP/IP Paket

Client-Process

Rechner A

Version	IHL	Type of Service	Total Length		
Identification			Flags	Fragment Offset	
Time to Live	Protocol=6 (TCP)		Header Checksum		
Source Address					
Destination Address					
Options				Padding	
Source Port			Destination Port		
Sequence Number					
Acknowledgement Number					
Data Offset		U	A	P	R
		R	C	S	S
		G	K	H	T
Checksum			Urgent Pointer		
TCP Options				Padding	
TCP Data (max. 64Kb)					

Netzwerk
LAN oder
Internet

Daemon

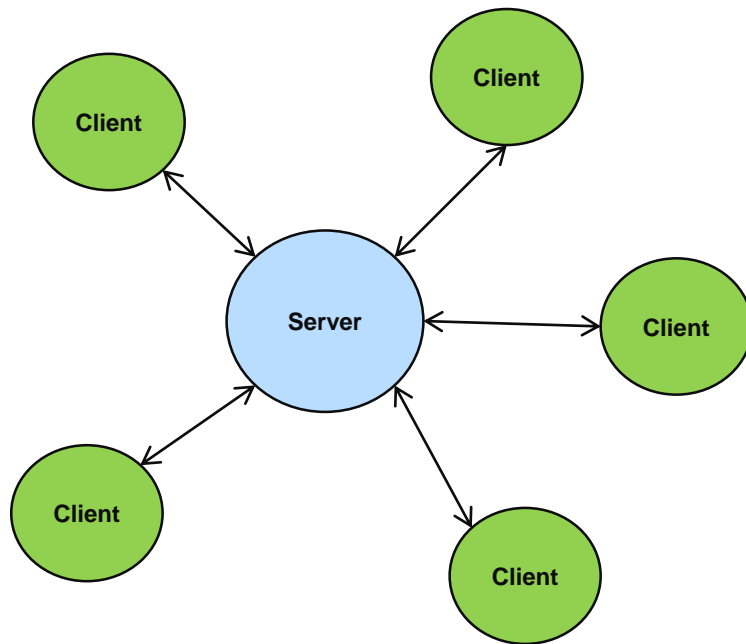
Rechner B

IP: 123.12.34.56

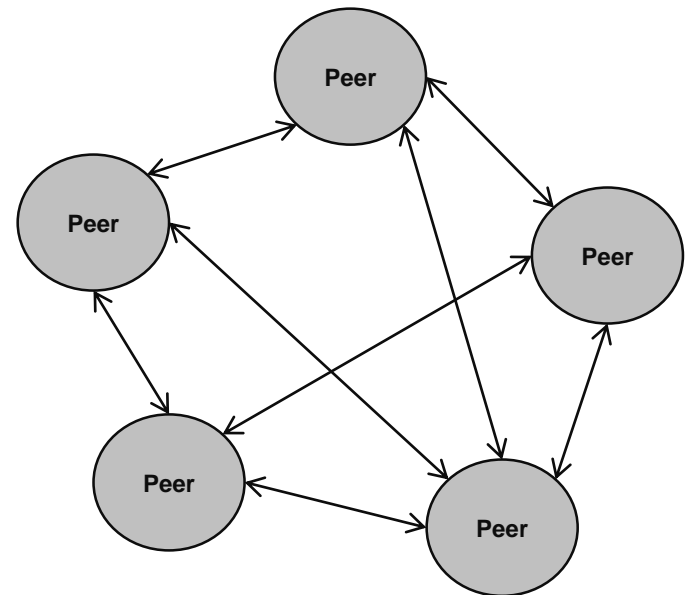
Port 3240

System Topologie

Client-Server (Master – Slave)



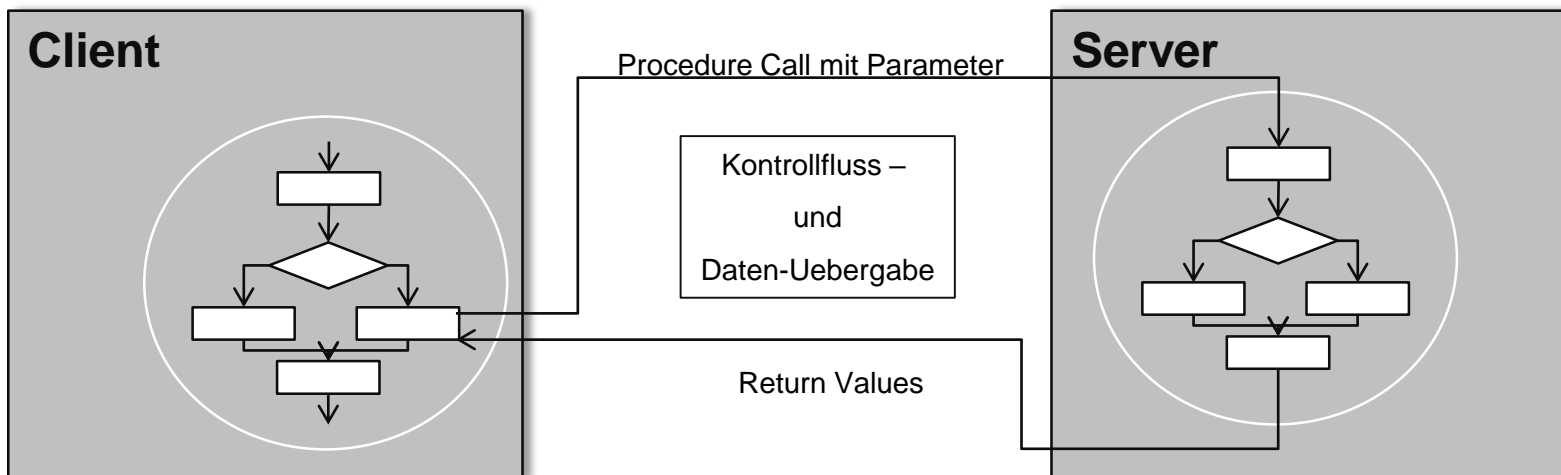
Peer to Peer (Gleichberechtigte Partner)



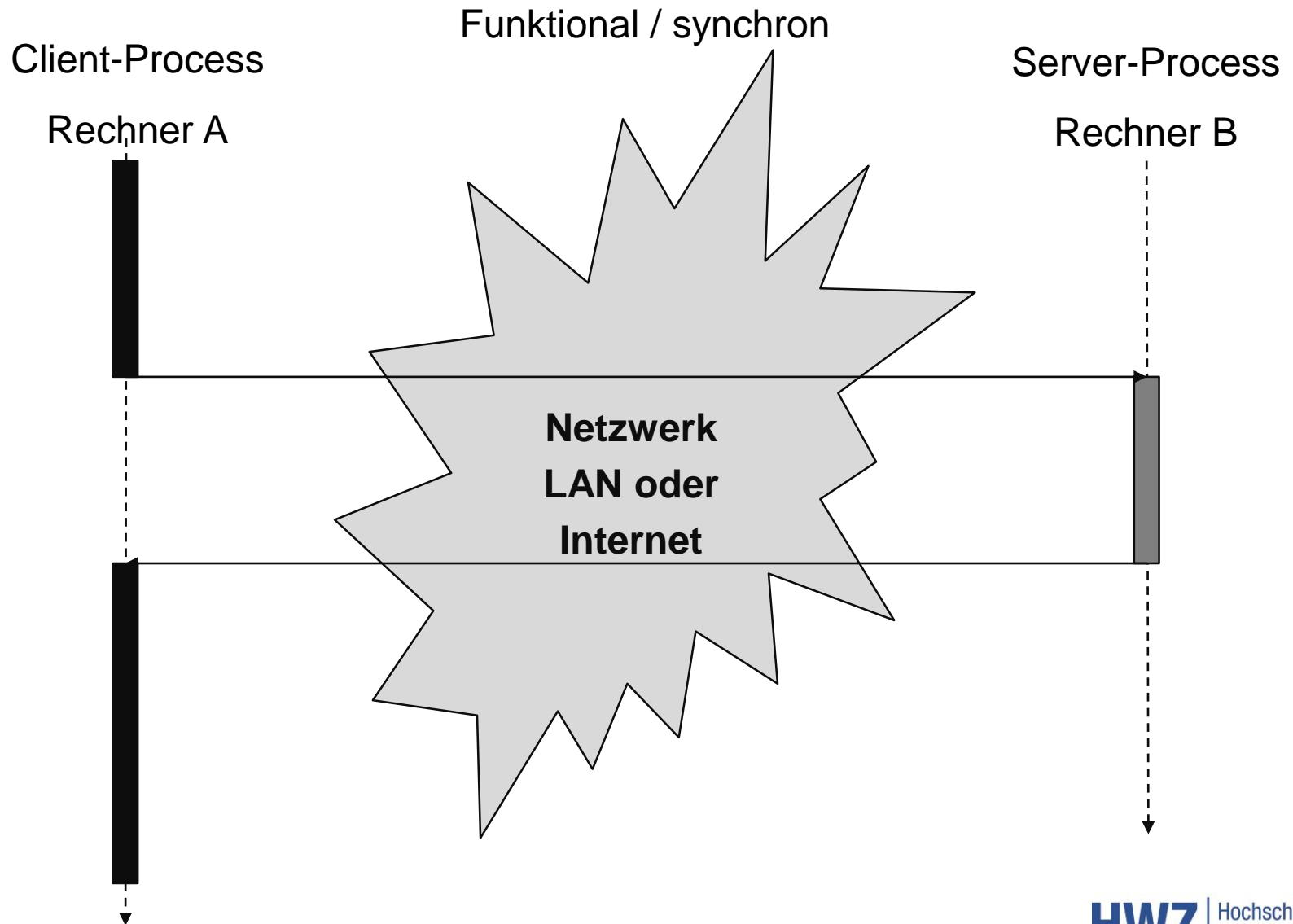
Systemarchitekturen

Client-Server:

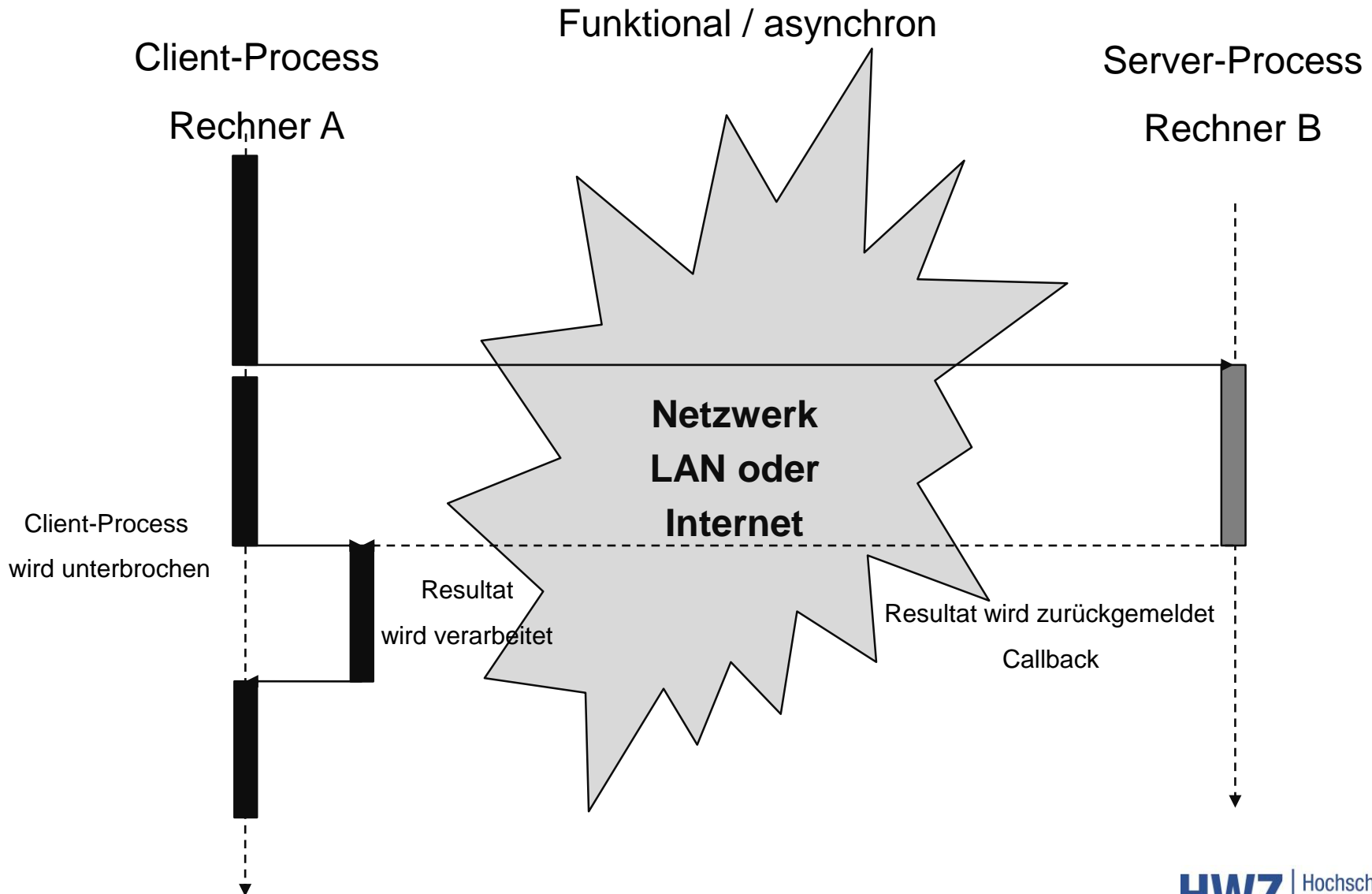
- Entfernte Prozeduraufrufe z.B. Remote Procedure Call (RPC)
- Grundsätzlich synchron (kein Ausnutzen von Parallelität)
 - ➔ Erweitert mit asynchronen Mechanismen (z.B. Callbacks)
- Hindernisse:
 - Datenflüssen / Konvertierung / Andere Adressräume / Andere OS
 - Server-Prozess muss aktiviert werden



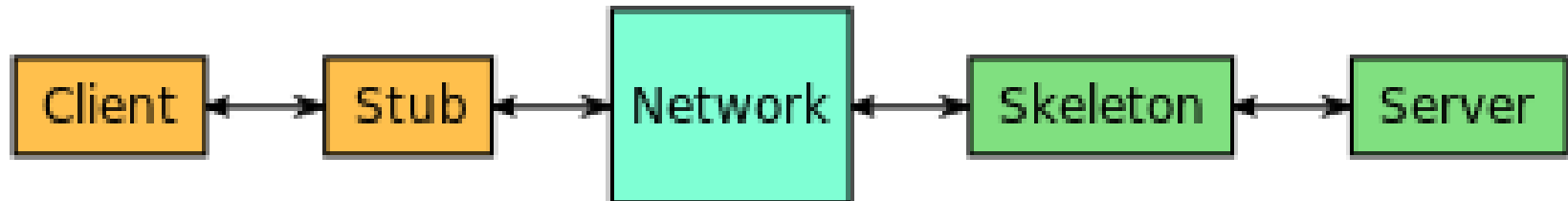
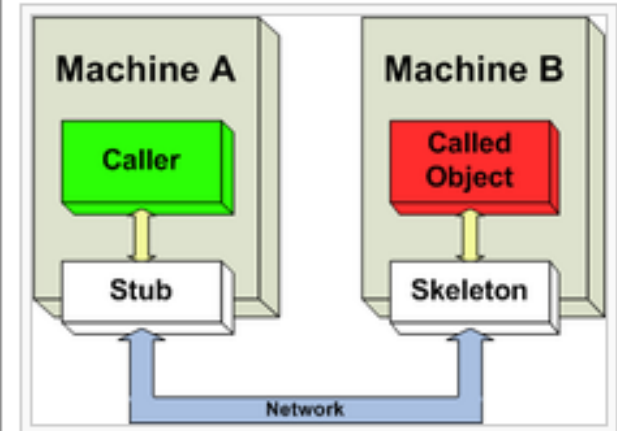
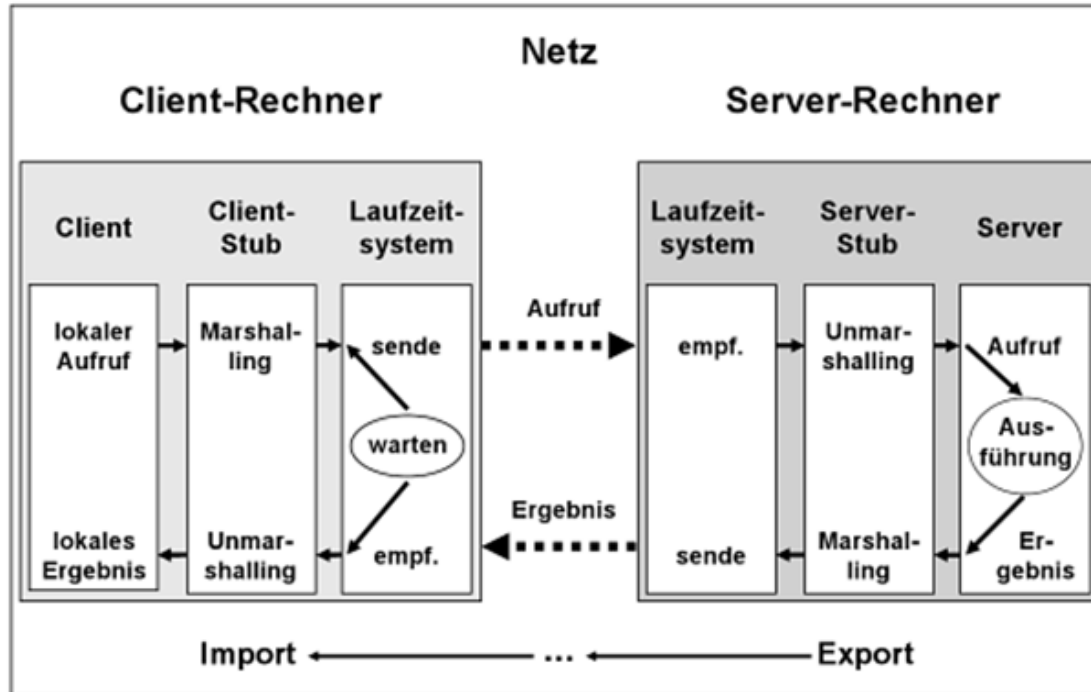
Remote Procedure Call (RPC)



Remote Procedure Call (RPC)



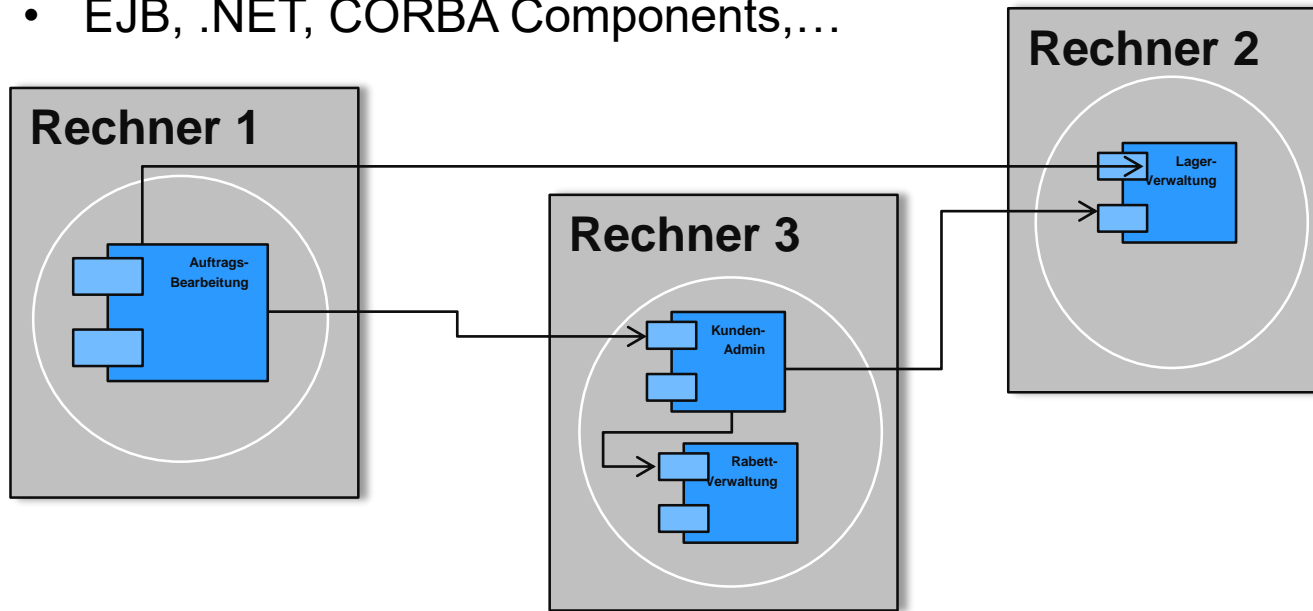
RMI (Remote Method Invocation)



Systemarchitekturen

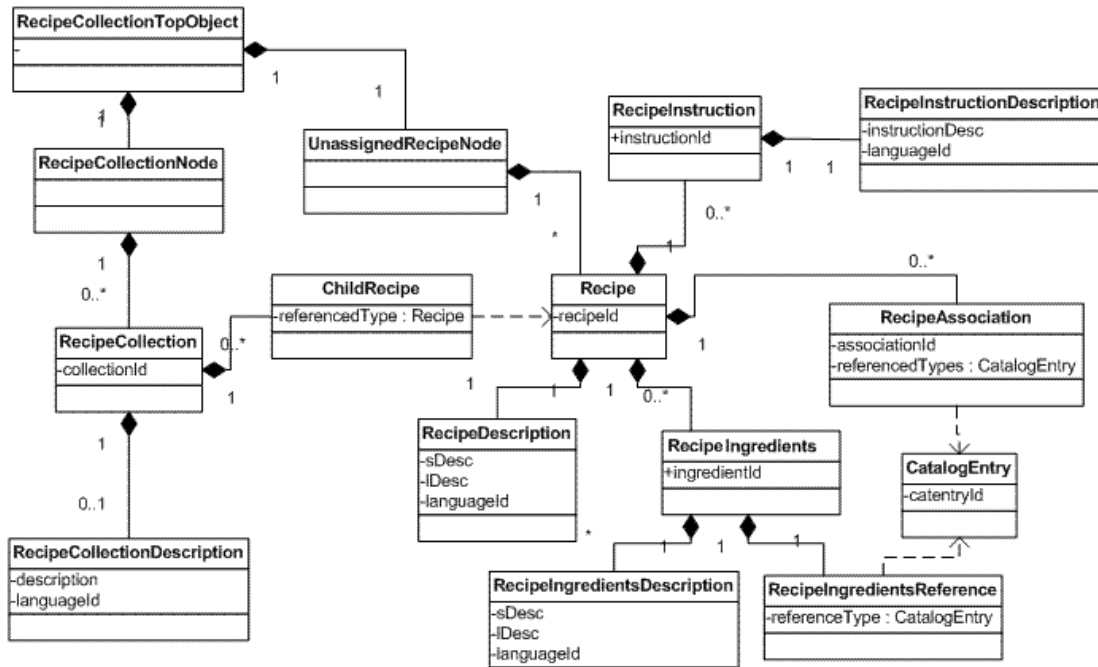
Component Based:

- Component erweitern Objektenmodell
 - Übergeordnete Business-Aufgabe
 - Allgemeine Aufgaben z.B. Persistenz, Transaktions,.. müssen nicht von der Komponente gelöst werden
 - EJB, .NET, CORBA Components,...



Distributed Objects:

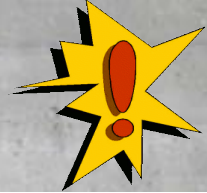
- Objekte sind verteilt (z.B. RMI (Java world), CORBA (Heterogen),...)
 - Methoden von «fernen Objekten» können aufgerufen werden
 - Objekte können als Parameter Methoden übergeben / return values werden



- Hindernisse:
 - Die Verteilung passiert während deployment oder zur Laufzeit
 - Unsichtbar für den Entwickler?
 - Objekte müssen gefunden werden (z.B. JNDI, Registry, IOR..)

Service Oriented Architectures (SOA) (Web-Services ist ein Spezialfall)

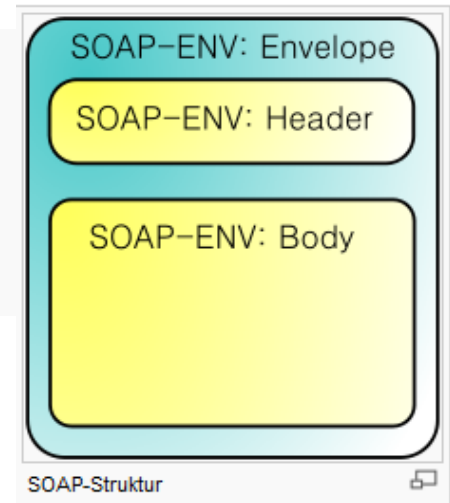
- Grobgranulare Bausteine werden angeboten, gesucht und genutzt
- Wohldefinierte Schnittstelle
- Hohe Verfügbarkeit
- Komposition möglich
 - Orchestrierung: Aus Diensten wird neuer Dienst zusammengesetzt
 - Choreographie: ein Geschäftsprozess wird aus Diensten kombiniert
- Kommunikationsprotokoll: SOAP (Simple Object Access Protokoll)
- Beschreibung des Services (plattform-, sprach- und protokoll-unabhängig):
 - WSDL (Web-Service Definition Language)
 - REST-API



Methode

```
<?xml version="1.0"?>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope">
  <s:Body>
    <m:TitleInDatabase xmlns:m="http://www.lecture-db.de/soap">
      DOM, SAX und SOAP
    </m:TitleInDatabase>
  </s:Body>
</s:Envelope>
```

Kriterien (Parameter)



Request ID

```
<?xml version="1.0"?>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope">
  <s:Header>
    <m:RequestID xmlns:m="http://www.lecture-db.de/soap">a3f5c109b</m:RequestID>
  </s:Header>
  <s:Body>
    <m:DbResponse xmlns:m="http://www.lecture-db.de/soap">
      <m:title value="DOM, SAX und SOAP">
        <m:Choice value="1">Arbeitsbericht Informatik</m:Choice>
        <m:Choice value="2">Seminar XML und Datenbanken</m:Choice>
      </m:title>
    </m:DbResponse>
  </s:Body>
</s:Envelope>
```

Antwort (Return Value)

Web-Services nutzen

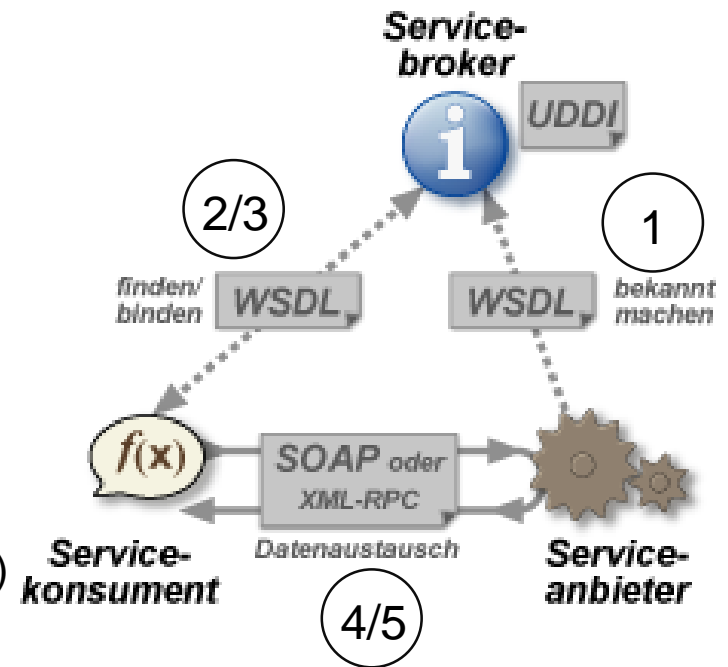
1. Veröffentlichen / Bekannt machen in einem Verzeichnis
2. Service suchen (abfragen)
3. Details über Dienst abfragen (Beschreibung)
4. Dienst nutzen (Service aufrufen)
5. Resultat verarbeiten

Glossar

SOA (Service Oriented Architektur)

SOAP (Simple Object Access Protokoll)

UDDI (Universal Description, Discovery and Integration)
(siehe www.uddi.org)



SOAP-Services nutzen in Python

```
import zeep

# Client config using WSDL
client =
zeep.Client(wsdl='http://www.soapclient.com/xml/soapresponder.wsdl')

# Calling an Soap Web-Service and display result
print(client.service.Method1('Zeep', 'is cool'))
```

SOAP-Services kreieren in Java (JEE)

```
package com.rothlin.webservice;

import javax.jws.WebMethod;
import javax.jws.WebService;

@WebService
public class FirstWebService {

    @WebMethod(exclude=false)
    public String ping() {
        return "Webservice (1.0.0.0)::FirstWebService:ping() called!!!!!!";
    }

    @WebMethod() // ctrl-space shows you the options
    public String sayHelloName(String name) {
        return "Hello " + name;
    }

    @WebMethod(exclude=false)
    public double calcCircleArea(double radius) {
        return radius*radius*Math.PI;
    }
}
```

SOAP-Services deployment / testing using Glassfish

The screenshot displays the Eclipse IDE environment with the Glassfish Server Open Source Edition interface. The left sidebar shows the Package Explorer with a project structure including '04_a_WebService_Simple'. The main window displays the 'Web Service Endpoint Information' for the selected service, providing details such as Application Name, Tester, WSDL, Endpoint Name, Service Name, Port Name, Deployment Type, Implementation Type, Implementation Class Name, Endpoint Address URI, and Namespace.

Web Service Endpoint Information

View details about a web service endpoint.

Application Name: 04_a_WebService_Simple

Tester: /04_a_WebService_Simple/FirstWebServiceService?Tester

WSDL: /04_a_WebService_Simple/FirstWebServiceService?wsdl

Endpoint Name: FirstWebService

Service Name: FirstWebServiceService

Port Name: FirstWebServicePort

Deployment Type: 109

Implementation Type: SERVLET

Implementation Class Name: com.rothlin.webservice.FirstWebService

Endpoint Address URI: /04_a_WebService_Simple/FirstWebServiceService

Namespace: http://webservice.rothlin.com/

The bottom status bar shows the Glassfish 4 server at localhost [domain1] is started and synchronized, with the 04_a_WebService_Simple service also synchronized.

SOAP-Services generated and published WSDL

```
+<portType name="FirstWebService"></portType>
-<binding name="FirstWebServicePortBinding" type="tns:FirstWebService">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  <operation name="ping">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
  <operation name="calcCircleArea">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
  <operation name="sayHelloName">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>
<service name="FirstWebServiceService">
  <port name="FirstWebServicePort" binding="tns:FirstWebServicePortBinding">
```

SOAP-Services generated Test Web-Application

Web Browser

http://desktop-nskdqsg:8080/04_a_WebService_Simple/FirstWebServiceService?Tester

FirstWebServiceService Web Service Tester

This form will allow you to test your web service implementation ([WSDL File](#))

To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

```
public abstract java.lang.String com.rothlin.webservice.FirstWebService.ping()
```

ping ()

```
public abstract double com.rothlin.webservice.FirstWebService.calcCircleArea(double)
```

calcCircleArea ()

```
public abstract java.lang.String com.rothlin.webservice.FirstWebService.sayHelloName(java.lang.String)
```

sayHelloName ()

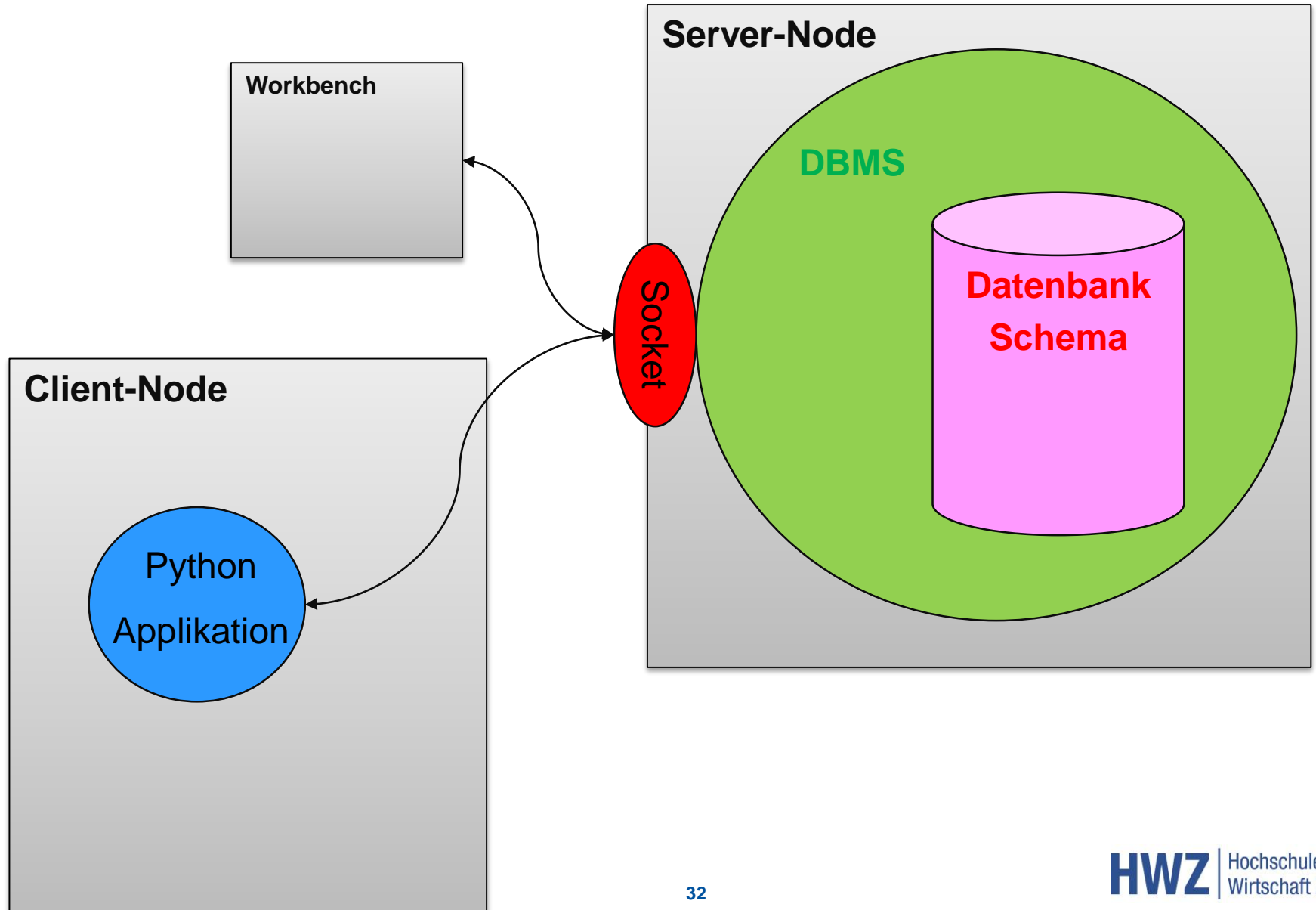
SOAP-Services nutzen in Python

```
import zeep

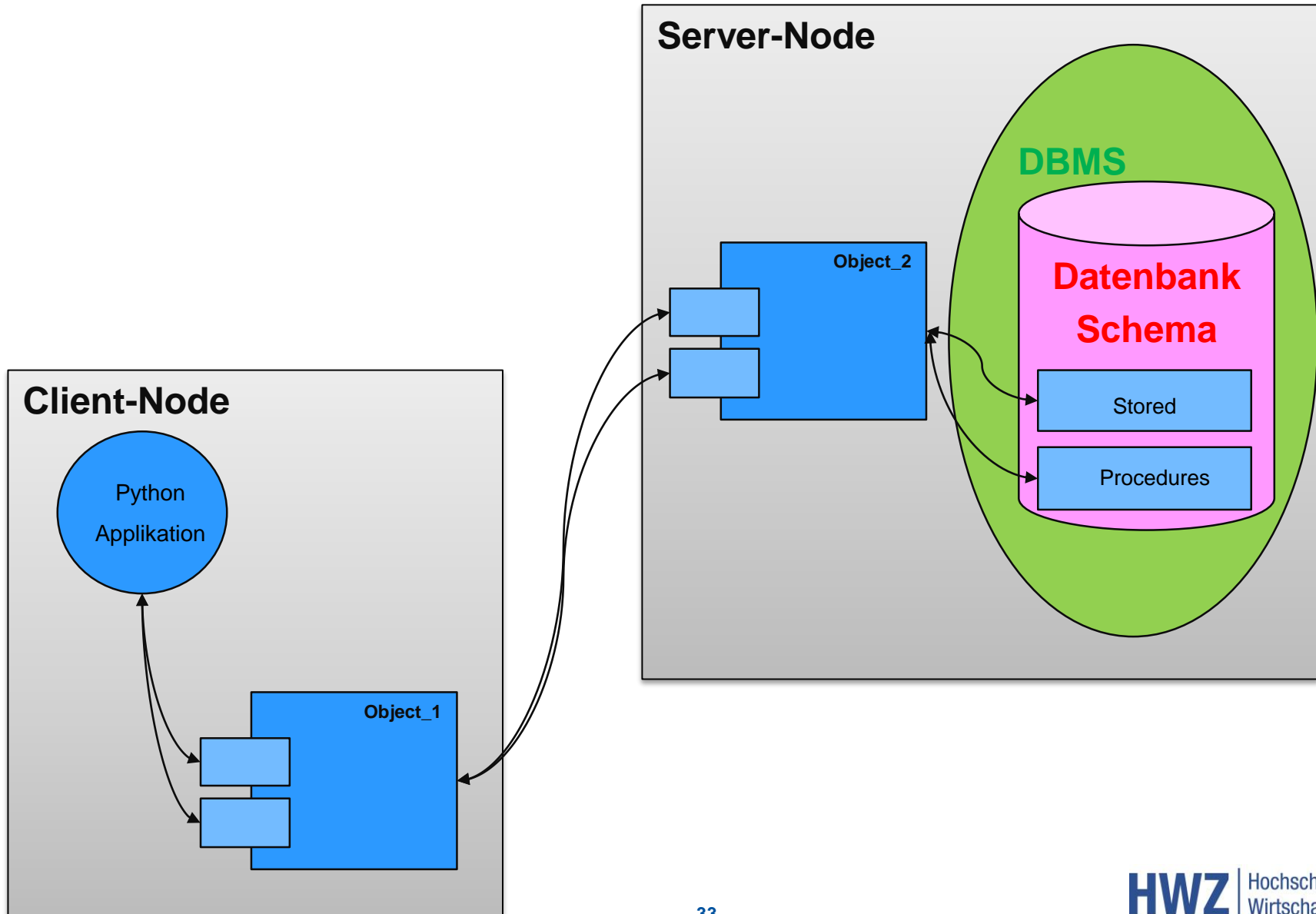
# Client config using WSDL
client = zeep.Client(wsdl='http://desktop-
nskdqsg:8080/04_a_WebService_Simple/FirstWebServiceService?wsdl')

# Calling Web-Services and display result
print("Pin() : ", client.service.ping())
print('sayHelloName("HWZ"):', client.service.sayHelloName("HWZ"))
print('calcCircleArea(5) : ', client.service.calcCircleArea(5))
```

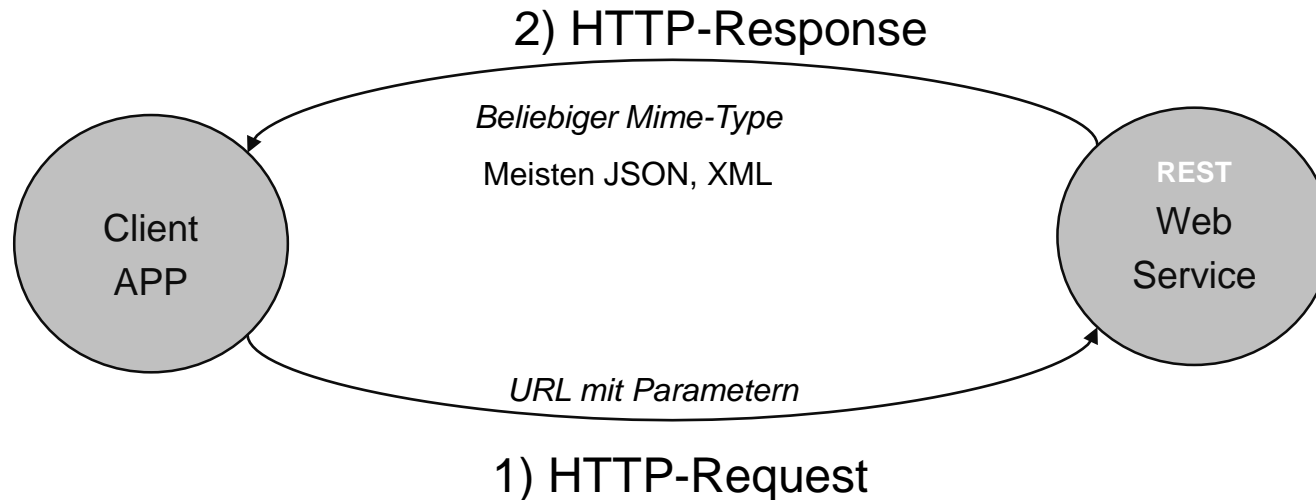
DB Anbindung in Python



Abstraktion-Level



REST Service



API Doc (Beschreibt Request und Parameter für Mensch)

1. Request kann mit Parametern direkt im Browser eingegeben werden (get)
2. Die Response kann ebenfalls dann direkt im Browser begutachtet werden

Beispiele von APIs:

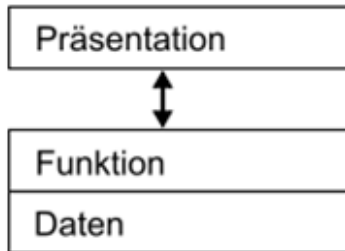
<https://api.openweathermap.org/>

<https://api3.geo.admin.ch/>

<https://tel.search.ch/api/help>

<https://www.webtimiser.de/google-maps-api-key-erstellen/>

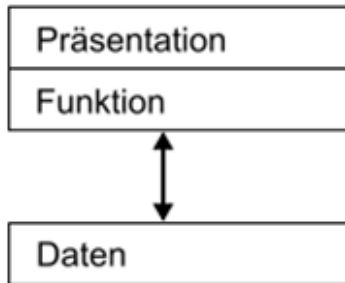
N-Tier (Ebene)



THIN-CLIENT, AKTIVER SERVER

- + zentrale Administration & Wartung
- + Kostenersparnis
- + Sicherheit (nur 1 Server muss geschützt werden)
- + Flexibilität

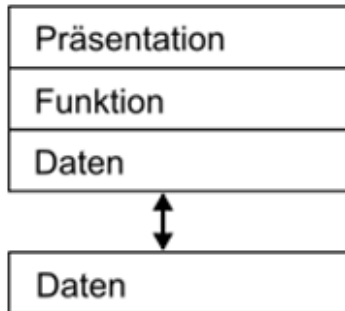
- hohe Belastung für Server



DATEN-SERVER

- Server liefert nur benötigte Daten
- + jeder Client kann selber rechnen → höhere Performance
- + zentrale Datenhaltung bleibt bestehen → Sicherheit

- hoher Installationsaufwand, da jeder Client alle Applikationen braucht



FAT-CLIENT

- + Entlastung Server & Datenleitung durch Plug-Ins
- + Weiterarbeiten möglich, wenn Kommunikation zum Server gestört

- lange Dauer, bis alle Clients Updates haben
- hohe Wartungskosten

N-Tier (Ebene)

Presentation tier

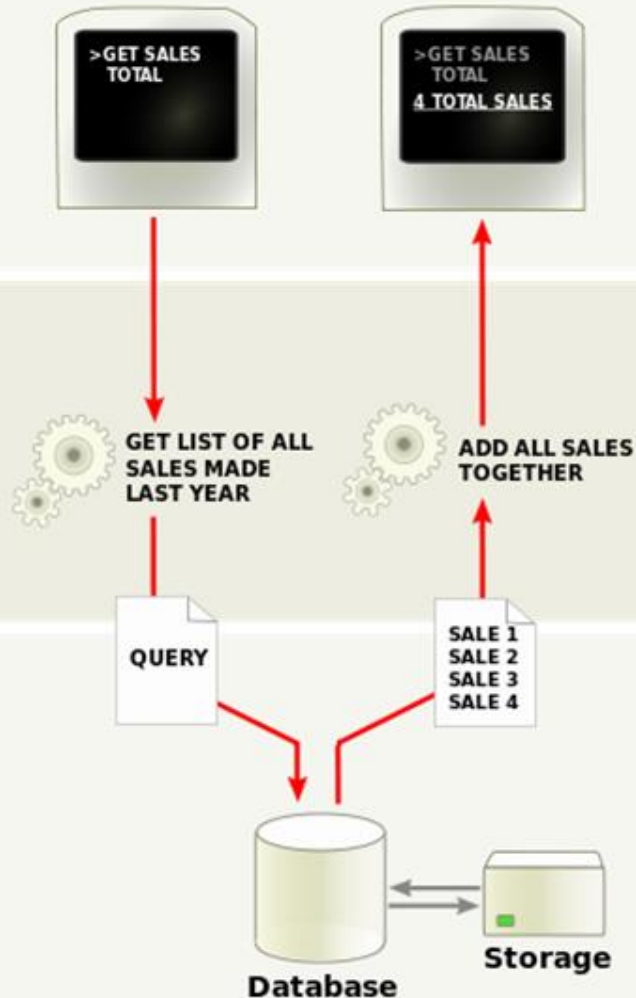
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

Logic tier

This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

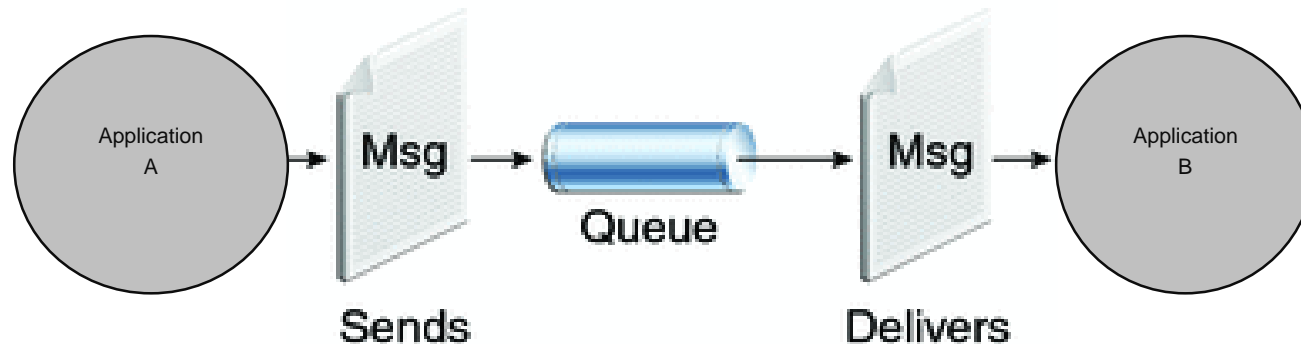
Data tier

Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.



business logic
logic tier
data access tier
middle tier

Message-Driven Architektur



Point-To-Point: Sender schreibt msg in eine Queue und **ein** Empfänger kann diese empfangen

Publish-Subscribe: Sender broadcasted Meldungen unter einem **Topic** (Domain). Mehrere Empfänger können sich auf diese Topic abonnieren und erhalten diese Meldungen

MDB Beispiel: <http://www.roseindia.net/ejb/example-of-messageBean.shtml>

Begriffe: Distributed Computing

Abrev	Begriff
SOA	Service Oriented Architecture
OMA	Object Management Architecture
CORBA	Common Object Request Broker Architecture
IOR	Interoperable Object Reference (eindeutige Object Reference im Internet)
IIOP	Internet Inter-Objectbroker Protokoll
SOAP	Simple Object Access Protokoll
RMI	Remote Methode Invocation (Stubs / Skeleton ; JAVA)
IDL	Interface Definition Language (Sprach Unabhängig) → IDL Compiler
WS	Web-Service WS (Soap)
REST	Representational State Transfer (Web-Service über URL-Request/Response)
API	Application Programming Interface